

Integrating Generative AI for Quality Engineering into the Software Lifecycle

Rozumenko, A., Udovychenko, A., King, T. M.

{ Artem_Rozumenko, Anastasiia_Udovychenko, Tariq_King }@epam.com

Abstract

Software practitioners and development teams are using generative AI and large language models like GPT-4, LLaMA, and PaLM for quality engineering and testing use cases. These include but are not limited to software requirements and design validation, test case and test script generation, and test result and coverage analysis. But how effective is generative AI in practice when it comes to delivering software faster without compromising quality? In other words, how do teams effectively apply this emerging technology in practical scenarios? In this paper, we describe an approach for integrating generative AI into the software development lifecycle. Our approach provides a holistic methodology that keeps the human-in-the-loop, while incorporating generative AI tools into communication channels, software hubs, integrated development environments, and test execution and reporting platforms. To keep our discussion grounded in practice, we present industrial case studies in which global teams have used an enterprise-ready generative AI collaboration platform for realizing software quality and productivity gains.

Biographies

Artem Rozumenko is a highly experienced technology leader and Director of Technology Solutions at EPAM Systems. With a strong foundation in software engineering and a deep passion for Performance Optimization, Security, and DevOps, Artem has built a distinguished career over the past decade. He began his professional journey as a manual tester and quickly advanced through roles in test automation, software development, and cloud operations. In his current role, Artem is heading up various Generative AI transformation initiatives within EPAM. He is based in Ukraine and continues to be an influential figure in the IT services and consulting industry.

Anastasiia Udovychenko is the Lead Software Testing Engineer with more than 10 years of experience in software quality assurance. Anastasiia is a technical leader with a customer-first mindset for achieving delivery excellence through lean governance, open communication, and agile thinking. She has extensive experience in QA across various domains, including e-commerce, financial services, e-resource management systems and laboratory information management systems. At EPAM, has been conducting demos, assessments and helping to educate others on AI for quality engineering. Her academic

background in Applied Mathematics, combined with industry experience, enables her to approach QA challenges with a unique analytical and problem-solving mindset.

Tariq King is a recognized thought-leader in software testing, engineering, DevOps, and AI/ML. He is currently the CEO and Head of Test IO, an EPAM company, and has formerly held positions such as VP of Product-Services, Chief Scientist, Head of Quality, Quality Engineering Director, Software Engineering Manager, and Principal Architect. Tariq has published over 50 research articles in peer-reviewed IEEE and ACM journals, conferences, and workshops. He has served as an international keynote speaker and trainer at leading software conferences in industry and academia and is co-creator of the Artificial Intelligence United's (AIU) certification in Gen-AI Assisted Test Engineering.

1 Introduction

A future where AI automates software engineering tasks is upon us. Generative AI (GenAI) and large language models (LLMs) are enabling higher degrees of software automation when it comes to requirements engineering, planning, project management, development, testing, and deployment. AI-assisted engineering tools are becoming more ubiquitous, easier to use, interact with, and incorporate into our everyday lives as software professionals. Many software practitioners, product owners, and researchers have been investigating ways to integrate GenAI into the software development lifecycle.

An area that continues to receive much attention when it comes to GenAI is quality engineering. Can we employ GenAI to accelerate software development? If so, would this result in better software? Or would any early productivity gains be short-lived due to rising defect rates? Speaking of quality, this technology also has a dark side. One in which AI infused applications, models, and assistants hallucinate. The non-deterministic nature of LLMs raises legitimate concerns around trustworthiness, reliability, security, and privacy, while the method by which these models give pause associated with intellectual property rights and ownership.

Although these technologies are still in their infancy, it is never too early to formalize approaches, tools, and techniques that promote the safe and responsible use and integration of AI. This paper presents a holistic approach for integrating GenAI into the software lifecycle, starting first with quality engineering activities. Quality engineering has been specifically selected as a means of aligning it with another important and related subject – that of productivity.

The remainder of this paper is organized as follows: the next chapter explores the relationship between productivity and quality. Section 3 discusses generative AI for quality engineering. Section 4 presents our proposed approach. Section 5 contains a case study that applies the approach in the context of a client-facing engagement. Finally, in Section 6 we conclude the paper and discuss future directions.

2 Productivity Versus Quality

Individuals, organizations and businesses are excited about the potential for AI to transform the world as we know it. Many believe that AI will revolutionize our personal and work lives by automating routine tasks and acting as a force multiplier in several industries. In the software industry, researchers and practitioners have been experimenting with the use of generative AI (GenAI) for software productivity [REF]. GenAI is being integrated into tools, processes and practices as a means of enhancing software engineering and other lifecycle activities. However, it is important to acknowledge that productivity in the context of the software lifecycle is not easy to define and can be extremely difficult to measure.

This section starts by describing general aspects of productivity and then focuses on the foundational principles necessary to quantify productivity during the software lifecycle. The main idea is that such a foundation is necessary to integrate generative AI for engineering into the software lifecycle. Furthermore, to ensure that quality is not compromised for speed when leveraging generative AI, we wrap up the section by emphasizing the direct relationship between quality and productivity due to the important role that validation plays in our proposed approach.

2.1 Defining Productivity

There are three general concepts that intersect to define productivity: **quantity**, **quality** and **efficiency**. When we think of being more productive, we tend to think about an increase in number or quantity of tasks being completed, typically over some unit of time. However, getting more things done faster only results in true productivity gains if those things are completed well. In other words, the finished work must meet a certain quality standard. Lastly, the effort of operating at this new level of output or throughput should not be wasteful, but instead should be cost efficient.



2.2 Quantifying Productivity in the Software Lifecycle

Although it may start with delivery, quantifying productivity during the software lifecycle goes beyond just output or flow. For example, traditional metrics like lines of code or number of commits can provide a basic measure of output, but do not capture quality or long-term maintainability. Process metrics like cycle time, which tracks the time from when work starts to when it is completed, can indicate a productive flow but are not the only factors. Other factors beyond the measurement of output and flow include:

- **Engineering Culture.** A culture where engineers feel safe to take risks, ask questions and report issues without fear of negative consequences promotes productivity.
- **Work-Life Balance.** Software productivity measures must account for sustainable work patterns, ensuring engineers are not overworked as this can lead to burnout and decrease long-term productivity.
- **Team Dynamics.** Effective collaboration within and across the team is key to productivity. Metrics like pull request turnaround times, number of cross team meetings, and feedback loops can provide such insights.
- **Quality and User Satisfaction.** Monitoring the number of bugs per unit of work can help gauge the quality of output. Collecting and analyzing user feedback and satisfaction metrics ensures that the team’s productivity aligns with delivering value.
- **Automation and Tooling.** Automated testing and continuous integration and deployment (CI/CD) pipelines can significantly improve software productivity by reducing manual tasks and allowing the team to focus on higher-value activities.

Due to the complexity of software engineering, it is important to take a multi-tiered view of measuring productivity that includes measuring at the individual, team, and value stream levels. Tables 1 and 2 provide an example of such multi-tiered measurement for software testing. In the context of leveraging generative AI for software productivity, measuring at multiple levels within a given activity such as testing, and across different activities like requirements engineering or design, lends itself to creating a more holistic and accurate view of the positive or negative impacts on productivity when applying AI/ML technologies.

Metric	Description	Additional KPI's or Metric Formulas
Test Development Velocity	Measures the number of tests created by a team member over time.	# Test cases developed by team member per period # Test scripts developed by team member per period
Test Result Analysis Effort	Measures the individual effort of a team member when analyzing test results.	Average # Hours spent on result analysis per person.

Table 1. Individual Metrics for Measuring Productivity During Software Testing Activities

Metric	Description	Additional KPI's or Metric Formulas
Test Development Velocity	Measures the number of test artifacts created over a given time period.	# Test cases developed per period # Automated test scripts developed per period
Test Coverage	Measures how well the application or domain covered by tests.	% Requirements covered by tests % Program statements covered by tests % Automated tests
Test Effectiveness	Measures the degree to which tests are capable of finding defects.	Defect Detection Efficiency
Test Result Analysis Effort	Measures the team effort for analyzing test results.	# Hours spent on result or failure analysis
Test Automation Effort	Measures combined effort required for creating and maintaining test automation scripts.	Test Development Velocity Test Automation Stability
Defect Leakage	Ratio of defects found and fixed in given phase to total defects found	Defect Detection Efficiency Defect Containment Efficiency

Table 2. Team Metrics for Measuring Productivity During Software Testing Activities

2.3 Quality Equals Productivity

As software teams attempt to move faster during the lifecycle, a common mistake is to cut corners on quality. However, time-savings from paying less attention to quality early in the lifecycle is generally short lived as bugs and other issues appear, resulting in costly rework. It may be tempting to think that since generative AI can produce lots of content quickly, we can just use that content “as-is” to get the job done. This a significant risk which can be mitigated in practice if teams assume a direct relationship between quality and productivity. For example, formulating the equation $Q = P$ as a testing mnemonic or mantra for the team can signify that time spent improving quality also increases productivity and vice-versa. This can help avoid some major pitfalls related some the key challenges associated with leveraging generative AI for software lifecycle activities.

3 Generative AI for Quality Engineering

This section presents our approach to integrating generative AI (GenAI) for quality engineering into the software lifecycle. It starts by identifying some of the key challenges organizations face when using generative AI for engineering, and then describes use cases that apply generative AI to quality engineering and software testing tasks.

3.1 Challenges

Integrating GenAI into the software development lifecycle presents a range of challenges that organizations must navigate to fully realize the full potential of these technologies.

- **Security and Intellectual Property Ownership.** A primary concern with LLMs is the uncertainty surrounding security and intellectual property ownership. Using LLMs increases the risk of data breaches and unauthorized access to sensitive information, while the origin of the generated content raises questions about who holds the rights to LLM outputs. These issues create legal and operational problems that can deter organizations from adopting AI-driven solutions.
- **Workforce Upskilling.** Another significant challenge is the need for upskilling within the workforce. As GenAI tools become more integrated into development workflows, it is essential for developers and other stakeholders to learn how to interact with these tools effectively. This involves not only gaining technical proficiency but also developing an understanding of AI processes, interpreting outputs accurately, and leveraging AI capabilities to drive innovation. Without upskilling, companies risk underutilizing these tools and failing to achieve the desired productivity gains.
- **Content Management.** Compounding these issues is the potential for GenAI tools to produce large quantities of software artifacts that may appear correct to the untrained eye. This can create a false sense of security as subtle errors or inefficiencies may go unnoticed until later stages of development, where they can become costly to address. The sheer volume of AI-generated artifacts can overwhelm traditional processes, making it essential for companies to have adequate tool support and streamlined review practices in place.
- **Validation and Verification.** The accuracy of AI-generated outputs remains a critical concern. Incorrect or poorly structured inputs can lead to flawed results, but even correct inputs do not guarantee optimal outcomes. The inherent unpredictability of generative AI necessitates careful monitoring and validation of outputs, adding complexity to the development process and requiring robust testing practices.
- **Productivity Measurement.** Measuring the productivity gains from using GenAI is a non-trivial task. While GenAI tools can accelerate the production of software artifacts, faster output does not necessarily mean better results. The quality, maintainability, and alignment of AI-generated outputs with the project goals must be carefully evaluated. Companies must develop nuanced metrics that account for both the efficiency and effectiveness of GenAI, ensuring that it genuinely enhances the development process rather than merely increasing the pace of production.

3.2 Use Cases

Upon the release of ChatGPT and other popular large language models and assistants, we surveyed over 300 practitioners and collected more than 1200 use cases on how they were using this technology for software development. Over 30% of the use cases collected were related to quality engineering or software testing activities. Table 3 is a summary of the categories of the collected use cases, coupled with supporting examples.







<p>Test Case Design and Development</p>  <p>Examples</p> <ul style="list-style-type: none"> Analyzing requirements and automatically generating tests to cover application or component functionality. Generating user acceptance tests, step and feature files. Analyzing source code and API's and automatically generating tests that target the program implementation. 	<p>Test Code Generation and Maintenance</p>  <p>Examples</p> <ul style="list-style-type: none"> Generating executable test scripts for automated unit, integration, and system-level testing to address both functional and non-functional testing. Migrating existing test scripts from one language, framework or platform to another. Updating test scripts as the application evolves.
<p>Test Case Maintenance and Management</p>  <p>Examples</p> <ul style="list-style-type: none"> Updating tests over builds to keep pace with changes. Identifying duplicate test cases to reduce redundancy and maintenance efforts. Converting tests from one format or test case management system to another. Simplifying complex tests to reduce the number of steps, improve readability and understandability. 	<p>Test Data Generation and Management</p>  <p>Examples</p> <ul style="list-style-type: none"> Automatically generating different types of test data based on a description of data characteristics or application fields. Converting test data from one format or database platform to another. Systematically updating or appending test data with new or modified values.
<p>Test Planning, Execution and Coverage Analysis</p>  <p>Examples</p> <ul style="list-style-type: none"> Generating comprehensive test strategies and plans. Including recommendations for testing techniques, tools, and frameworks. Prioritizing, scheduling, or optimizing tests for execution. Identifying gaps in test coverage for a system or domain 	<p>Test Result Analysis and Defect Management</p>  <p>Examples</p> <ul style="list-style-type: none"> Automatically summarizing result of bug reports Grouping similar features or identifying common relationships or possible root cause for failures. Automated failure triage including categorization and severity assignment

Table 3. A Summary of Practitioner-Based GenAI for Quality Engineering Use Cases

By automating and optimizing critical quality engineering and testing processes—from test case design to defect management—AI not only has the potential to enhance software productivity but may also promote more thorough and reliable software testing. It is because of this high potential for transformation that we have formulated an approach and implemented supporting tools and platforms. Some of the supporting tools and platforms are freely available as part of our many contributions to open-source software, while others are integrated into EPAM service offerings. Regardless of their software licensing model, it is the hope of the authors that sharing the details of these projects will encourage others to innovate in this rapidly growing area of emerging technology.

4 Proposed Approach

Integrating the use of GenAI for quality engineering into software lifecycle requires careful planning, coupled with a robust infrastructure and focus on both technical and human-centric elements. To ensure a successful and sustainable integration, it is essential to follow a structured approach that incorporates secure access to AI models, solid engineering practices, seamless user experience, content management, and collaborative tooling. This section presents our proposed approach to integration which has evolved from our experience applying generative AI to QE tasks on a variety of client-facing projects. To keep our description rooted in practice, where applicable we provide references to products, platforms, frameworks, and accelerators that can be used to support the approach.

4.1 Secure Access to Enterprise-Ready Generative AI Models

Safeguarding corporate data and assets is paramount and therefore one of the first steps in our approach seeks to address concerns around security and data privacy when using GenAI in the software lifecycle. To ensure a high level of security, businesses need access to enterprise-ready AI models and infrastructure that offer full and exclusive control to the organization. This includes the ability to host these solutions privately, providing an added layer of safety and customization. This provides key assurances in the setup including making sure that prompts, completions, embeddings, and training data are completely isolated and secure. These assets are not shared with or accessible by other clients, nor are they available to model providers like OpenAI. The goal is that your proprietary data remains entirely within your control, safeguarded from external access or usage.

4.1.1 Proxy Implementation

One method for achieving this in practice is to implement a proxy that allows secure access a diverse array of AI models. This proxy serves as a centralized gateway, ensuring that teams can access and leverage the most suitable AI models for their specific testing needs without compromising security. Such a setup should allow for dynamic selection and switching between different models, depending on the task at hand. For example, open-source models might be preferred for certain types of analysis due to their transparency, adaptability and data security, commercially available public models have a huge knowledgebase and can be used to solve majority of the tasks where data privacy is less of a concern, proprietary models could be leveraged for their advanced capabilities and fine-tuned accuracy. The proxy ensures that all interactions with these models are secure, compliant with regulatory standards, and auditable.

4.1.2 AI DIAL

Merging the power of LLMs with deterministic code can facilitate the development of a unified interface for empowering businesses to leverage a spectrum of models, assistants, and more. This is exactly the philosophy behind the AI-powered Deterministic Integrator of Applications and LLMs (AI DIAL).

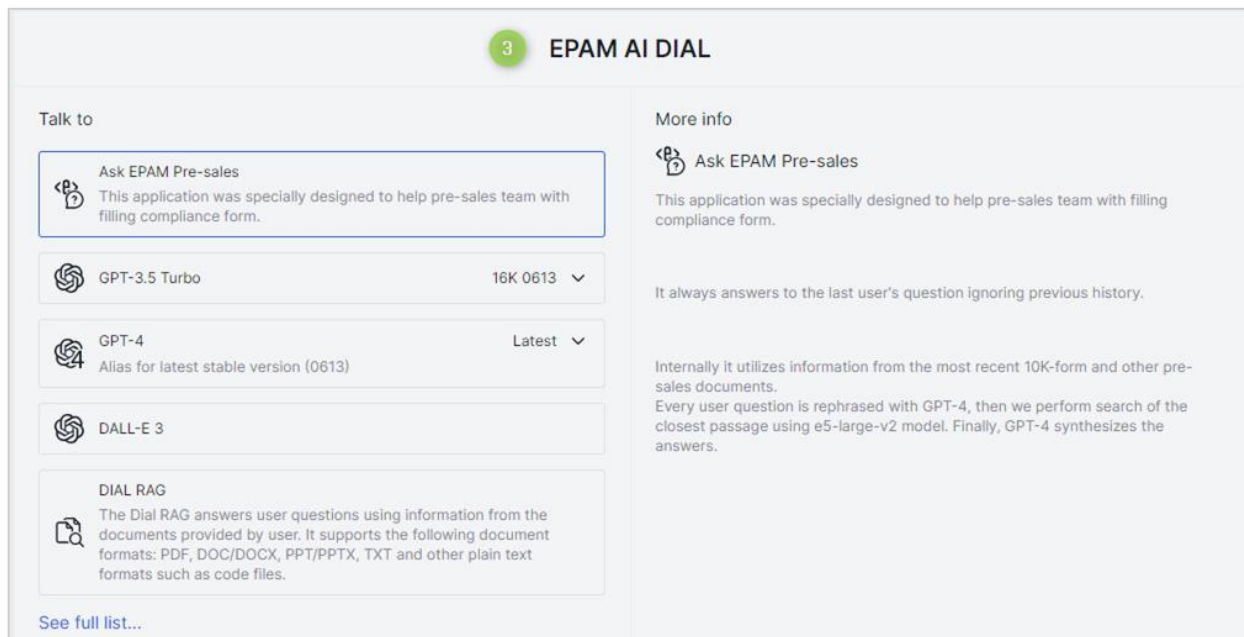


Figure 1. User Interface of Open-Source Chat-Based Accelerator Built for AI DIAL

Figure 1 is a screen capture of a chat-based accelerator built on AI DIAL. It allows the user to select which model or models they want to interact with and even provides a feature for sending the same prompt to two different models so that the results can be compared in real-time. Custom models, assistants or other applications can also be accessed via this interface, which is backed by a secure proxy. The type of integration and unified access provided by AI DIAL promotes the development of novel, AI-based, enterprise assets that co-exist seamlessly with an organization's existing workflows. In keeping with our long-standing commitment to open source, EPAM has released this chat-based accelerator under the Apache 2.0 licensing scheme. AI DIAL encourages responsible use, community innovation, and adoption of responsible AI standards within the industry.

4.2 Building on A Foundation of Engineering Excellence

Our approach recommends that the integration of GenAI into the software lifecycle be built on a foundation of engineering excellence. Engineering Excellence refers to the pursuit of the highest standards in engineering practices, encompassing the consistent delivery of high-quality, reliable, and innovative solutions that meet or exceed user and business expectations. It involves a commitment to continuous improvement, adherence to best practices, and the application of rigorous methodologies that ensure the optimal performance, scalability, and security of products and systems. Engineering excellence embodies a culture of transparency, accountability, collaboration, and ethical responsibility, driving teams to not only achieve technical precision but also to contribute meaningfully to the advancement of the engineering discipline and organizational success.

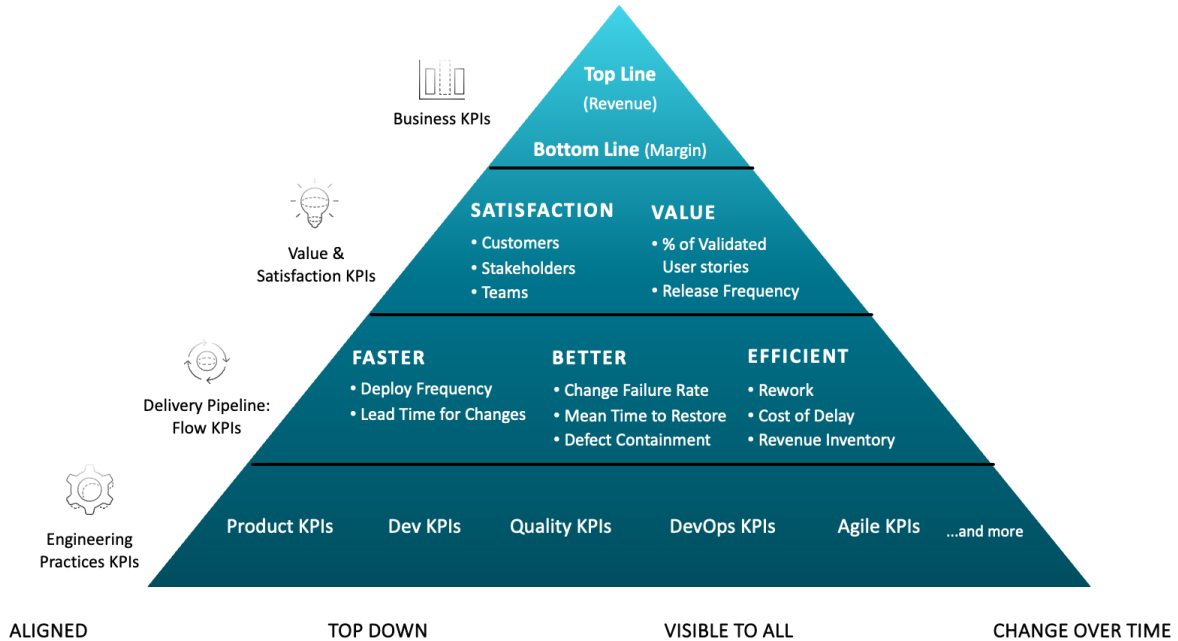


Figure 2. Engineering Excellence Key Performance Indicator (KPI) Pyramid

A key benefit of leveraging engineering excellence within our approach is that it promotes a strong, KPI-driven culture of engineering best practices. Figure 2 illustrates the various kinds and levels of Key Performance Indicators (KPIs) that a focus on engineering excellence can provide. Leveraging such a framework is particularly useful during the integration of GenAI as it supports conducting observations, measurements, baseline assessments and evaluating the impact of AI on quality and productivity at multiple levels. For example, before introducing AI into testing activities, it may be essential to capture baseline measurements on test execution times, defect rates, and test coverage. These baselines provide a point of comparison to directly measure the impact of AI integration on testing but can also be extended up to higher levels of the pyramid to gain insights into the overall value.

In cases where capturing baselines is not feasible, running concurrent experiments—where one set of testing activities is assisted by GenAI, and the other does not—can help assess the overall benefits and challenges. This may require the implementation of observability tools that monitor AI-driven processes in real time, ensuring that any deviations from expected behavior are promptly identified and addressed. Measurability, coupled with these observability practices, provides the data needed to refine AI models and continuously improve their performance in the software development lifecycle.

4.3 Seamless Integration into Existing Tools and Workflows

One of the most critical aspects of integrating GenAI into the software lifecycle is ensuring that it fits seamlessly into existing tools and workflows. Rather than requiring teams to adopt

entirely new tools or platforms, AI should be embedded within the products and systems they are already using. This approach minimizes disruption and reduces the likelihood of resistance from teams who may be hesitant to learn new technologies. For example, AI-powered test case generation, automation, and analysis should be integrated into the same IDEs, test management systems, and reporting tools that teams are already comfortable with. This integration allows AI to enhance the existing workflows without imposing additional cognitive load on the users, thereby increasing the adoption rate and ensuring that AI becomes a natural extension of their daily work. The remainder of this subsection provides concrete examples of such integrations in the context of quality engineering.

4.3.1. AI Jeannie (JIRA Plugin)

JIRA users currently experience inefficiencies and inconsistencies in their workflows, negatively impacting the quality of their sprints and issues. These inefficiencies often result in increased manual effort. Furthermore, there is a lack of seamless collaboration between Business Analysts and other JIRA users due to inconsistencies in language and terminology used for JIRA issues.

AI Jeannie is a freely available, open-source, plugin for JIRA that helps Business Analysts to create quick epic descriptions, user story descriptions, and acceptance criteria based on Project Definition configured. As illustrated in Figure 3, it works by allowing you to bring your own AI or LLM provider and helps users to quickly generate high-quality, accurate, and relevant requirements directly within JIRA. Sequence diagrams can also be automatically generated to allow stakeholders to validate them as part of their agile process and workflow.

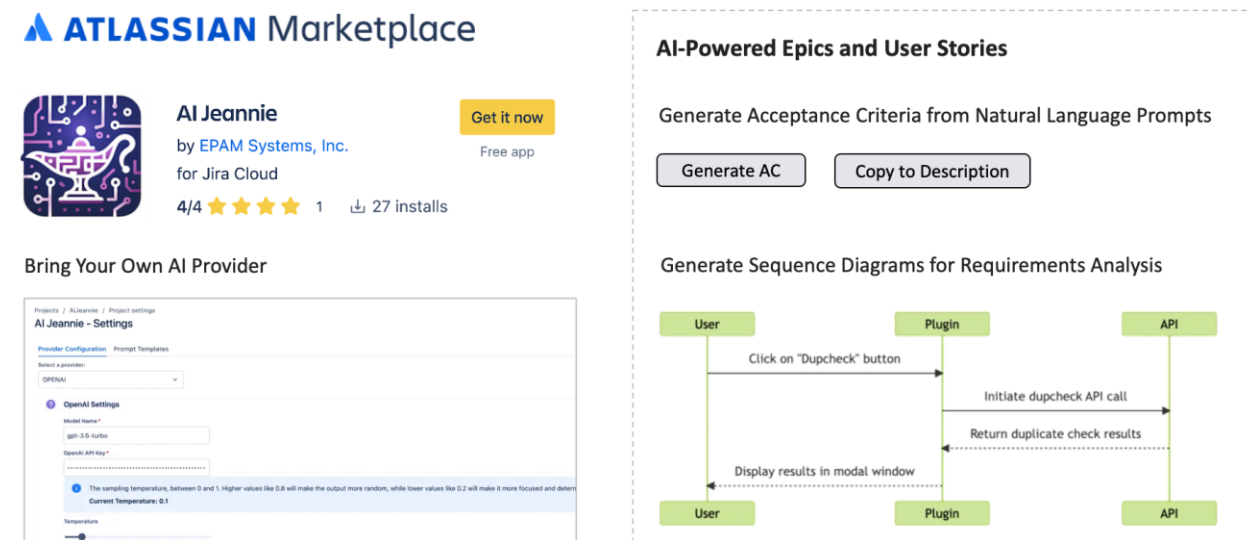


Figure 3. AI Jeannie JIRA plugin supports requirements-based quality engineering

4.3.2. Alita Code (IDE Plugin)

Software and test automation engineers often encounter challenges during development when implementing program-based tests. Generative AI and large language models perform quite well at code generation tasks, including creating automated test scripts. However, ultimately this code will live in a repository and be updated and maintained using an integrated development environment (IDE). Therefore, it makes sense that these types of tools be provided as extensions to the IDE, rather than outside of the programming environment.

Alita Code is an IDE extension seeks to improve the way that software engineers and testers develop, test, and maintain program code. It leverages AI to generate automated unit, integration, and system-level tests. It can also automatically add comments to code to make it more understandable and maintainable for engineering teams and provide AI-powered code suggestions. At the time of writing, the Alita Code plugin is available for multiple IDEs including Visual Studio Code and IntelliJ and can be configured for public or private LLMs. Figure 4 shows the Visual Studio Code variant, which exposes the following extension commands:

- **Init** – Initializes the plugin and creates a local prompt library folder in the open workspace.
- **Create Prompt** – Adds a new prompt to the local prompt library.
- **Extend Context** – Facilitates expanding the context of a given prompt in the local library.
- **Predict** – Presents user with a list of prompts for performing GenAI engineering tasks.
- **Similarity** – Presents user with a list of embeddings to run similarity search against.
- **Sync External Prompts** – Allows users to sync prompts to and from a shared backend.

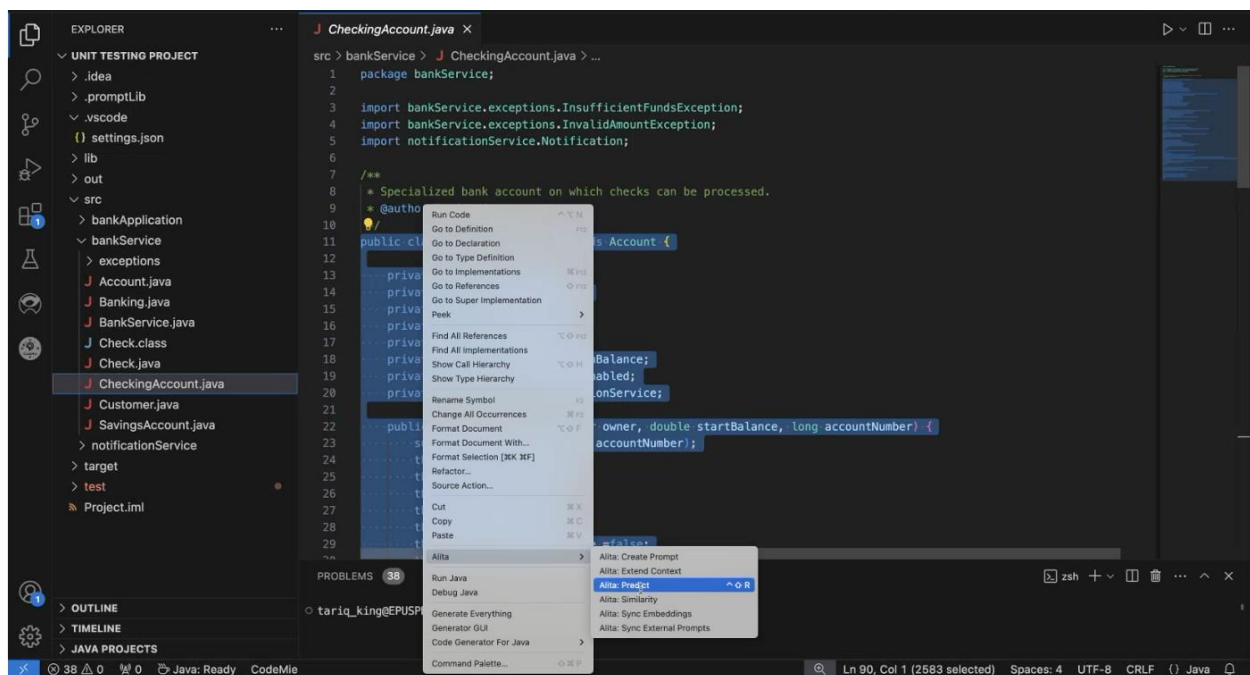


Figure 4. Alita Code Plugin for Integrated Development Environments

4.4 Supporting Content Management and Validation

AI-generated content, such as test cases, scripts, and reports, must undergo review and validation processes to ensure accuracy, relevance, and compliance with organizational standards. This involves setting up review pipelines where AI-generated outputs are automatically routed to human reviewers for approval.

Building connectors to existing test management systems and reporting tools is also essential. These connectors facilitate the seamless flow of AI-generated content into established systems, enabling teams to track, manage, and validate AI outputs alongside traditional testing artifacts. By making content management and validation an integral part of the AI integration process, organizations can maintain high standards of quality and reliability in their testing practices.

4.5 Facilitating Collaboration and Experimentation

Collaboration is a key ingredient for harnessing the full potential of GenAI. Mechanisms that enable teams to share prompts, refine models, and collectively improve engineering processes must be put into place. Such tools should support real-time collaboration, such as sharing environments or communicating via integrated chat and feedback systems. The proposed approach therefore emphasizes that beyond the technological elements, organizations must invest in practices aimed at helping teams work together more effectively with each other, and with AI.

In addition to a collaborative space for sharing, there should be private spaces for conducting localized experiments so that teams can explore new ideas with GenAI in a controlled environment. For example, in the early stages of experimentation it may be necessary to set up an isolated test environments where AI can be trialed without impacting the broader development or testing effort. Such experimentation fosters innovation and ultimately contributes to building a culture of continuous improvement.

4.6 Bringing it All Together

At EPAM and Test IO, we have applied the various elements of the proposed approach successfully and have used the content described here as guiding principles in developing our own AI collaboration platform. Our platform sits behind a secure proxy which provides access to a plethora of LLMs, and we have been using it to seamlessly integrate GenAI into testing and software engineering tasks. Wherever possible, ELITEA is used to help augment existing tools, frameworks, and process workflows with AI capabilities. Figure 5 contains a visual of the approach and illustrates the kind of bridge ELITEA provides to teams as an AI collaboration platform.

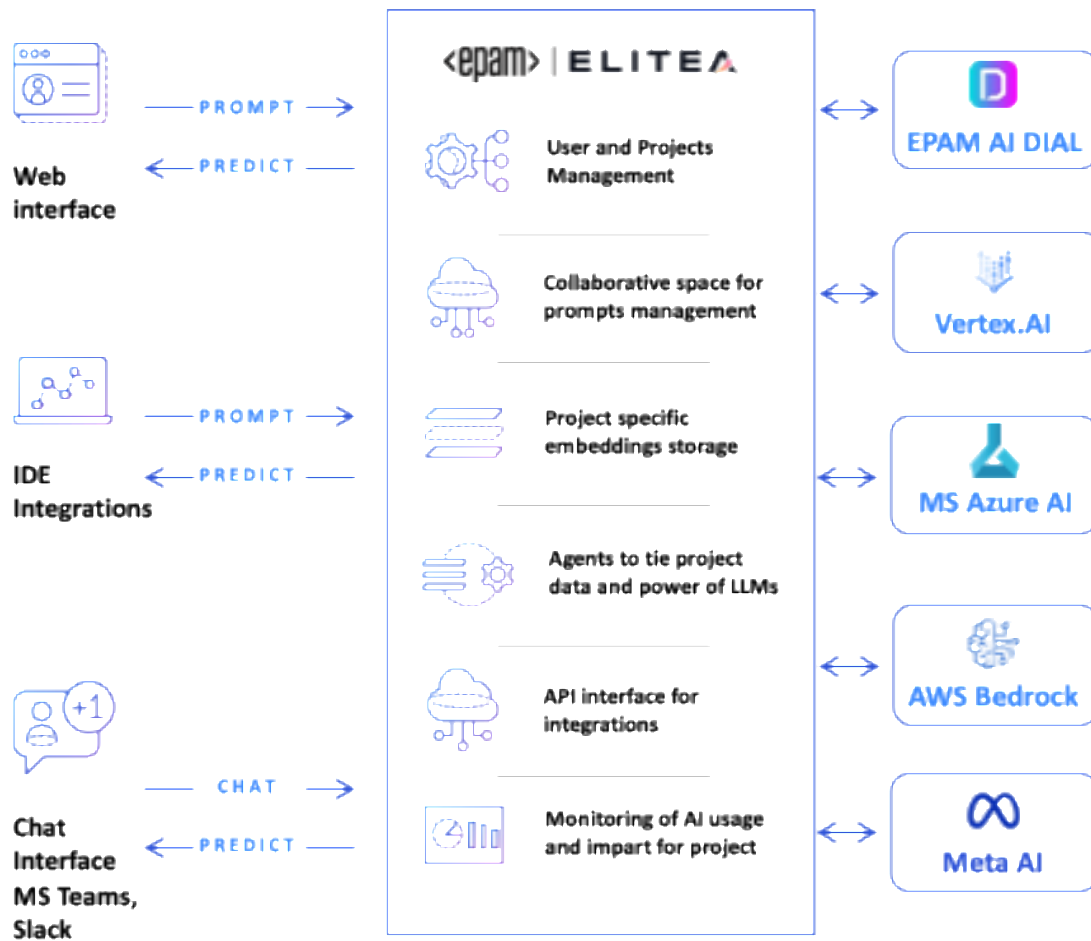


Figure 5. Seamlessly Integrating GenAI for Software and Quality Engineering with ELITEA

5 Case Study

The anonymized case study presented in this section is taken from a past client-facing, industry engagement that used the proposed approach to accelerate quality engineering and testing activities. It includes a description of the client, the problem motivation, key highlights of the engagement, and any results, findings or conclusions.

Reducing Testing Debt in Library Services with Gen AI

Client is a co-creator and key contributor of a cloud-based, open-source, library services platform. Although the platform uses a modular architecture that lends itself to test automation, the rapid pace and community-driven nature of the project has led to gaps in test automation. The growing reliance on manual testing motivated the client to reach out to us for guidance on leveraging GenAI for UI test automation to tackle the accumulated software testing debt.

Project Goals

The goal set for the project was to successfully integrate GenAI into the test script development workflow of the team members and automate a large number of end-to-end, manual UI tests within a 4-month period. Achieving this should in turn accelerate the release cycle due to the increased test automation.

Highlights

Some key highlights of the engagement included the use of crowd-sourced test automation contributors; a template generator; IDE plugin' and CLI tool all as GenAI usage enablers. In addition, there were dedicated engineers to conduct GenAI experiments using different prompts, models and approaches, and the Alita Code extension enabled team members to build and execute prompts within the IDE.

Results

Within the 4-month period, 1000+ end-to-end UI tests were automated. After a short ramp-up period, a subset of team members was onboarded with the GenAI skills and tools necessary. Figure 6 provides a snapshot of some of the project results. There was approximately a 15% increase in the test development velocity when using GenAI, and an estimated 20% time savings and close to 40% reduction in implementation costs. Another finding worth noting was that the seniority index of GenAI assisted engineering group was a few points lower than the non-AI group. In other words, a group with less senior engineers using AI, outperformed a group of more seasoned engineers without AI assistance.

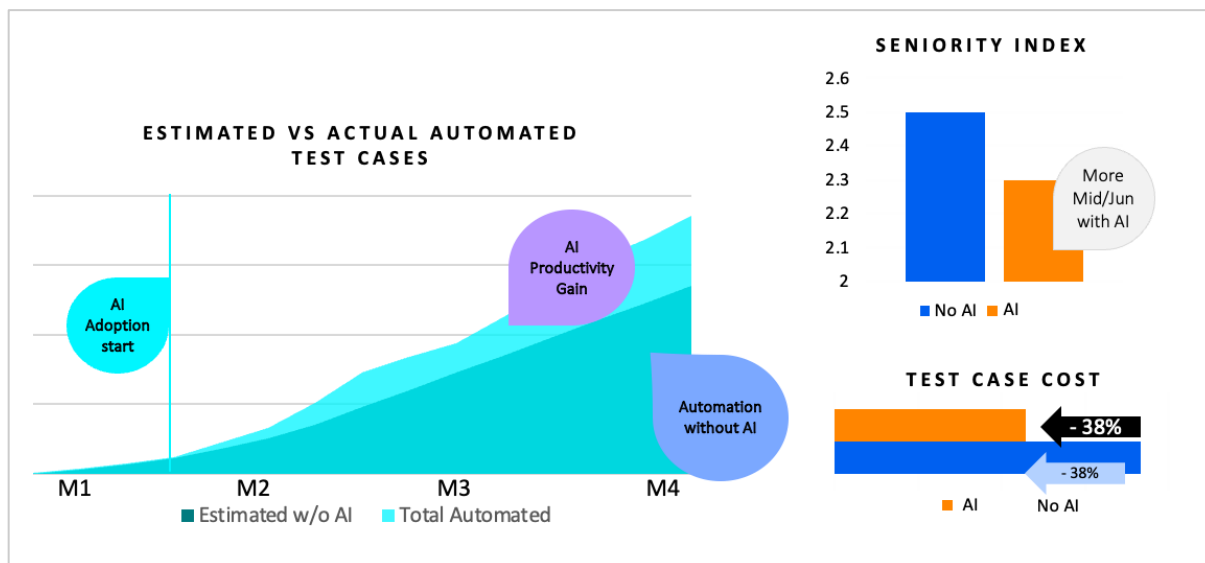


Figure 6. Snapshot of the Case Study Results

6 Conclusion

As demands on time to market for high-quality software grow, it is becoming necessary to accelerate the software pipeline. Consequently, the integration of generative AI into the software development lifecycle is changing how software is designed, tested, and delivered. This paper has explored the challenges and opportunities of embedding generative AI within software engineering processes, particularly focusing on quality engineering and testing. The integration of generative AI brings forth challenges such as security concerns, intellectual property issues, and the need to upskill the workforce. The temptation to prioritize speed over quality, especially with the rapid output capabilities of generative AI, is a common pitfall. However, the presented approach and associated case study are providing early indicators that such an integration provides a net benefit to the software development cycle. Still there is much more work and practical studies that are needed before we can draw any firm conclusions around the transformative potential of this technology.

Acknowledgements

The authors would like to thank the following people and the members of EPAM's Testing Competency Center for their contributions to this work: Ihar Bylitski, Adam Auerbach, Pavel Seviaryn, Matthew Gorelik, Viktoryia Valakh and Dmitry Tovpeko. We would also like to express our sincere appreciation to the entire EPAM and Test IO community. Their collective wisdom, experience, and collaborative spirit have enriched this work in countless ways, ensuring its relevance and impact on the industry.

References

"Continuous Testing and Quality Gates: A Practical Guide." *Journal of Software Testing*, Vol. 14, 2019.

Amershi, S., Cakmak, M., Knox, W. B., & Kulesza, T. "Power to the People: The Role of Humans in Interactive Machine Learning." *AI Magazine*, Vol. 35, No. 4, 2014.

Mitchell, M. *Artificial Intelligence: A Guide for Thinking Humans*. Farrar, Straus and Giroux, 2019.

Russell, S., & Norvig, P. *Artificial Intelligence: A Modern Approach*. 4th ed., Pearson, 2021.

"ReportPortal: An Open-Source Reporting Platform with Machine Learning Capabilities." Official Documentation, 2022. <https://reportportal.io/docs/>

"ELITEA: Platform designed to streamline the management, development, and collaboration of LLM assets." Official Documentation, 2024. <https://projectalita.ai>

"Alita Code: AI-powered IDE extension for developing, testing, and code maintenance." 2024.
<https://marketplace.visualstudio.com/items?itemName=ProjectAlita.alitacode>

"AI Jeannie: Jira extension to create Epics and User Stories using Generative AI." 2022.
<https://marketplace.atlassian.com/apps/1232950/ai-jeannie>

"AI DIAL: An Open-Source AI Orchestration Platform and Development Studio" 2024.
<https://dialx.ai/>

Whittaker, J. A., Arbon, J., & Carollo, J. *How Google Tests Software*. Addison-Wesley Professional, 2012.

Gumeniuk, D., & King, T. Improving Enterprise Scale Test Automation with ML-Based Predictive Analytics. In the Proceedings of the 2023 Pacific Northwest Software Quality Conference, October 2023.

One Hundred Year Study on Artificial Intelligence (AI100). 2016 Report, 13.
ai100.stanford.edu/2016-report

Kelly, K. "The Three Breakthroughs That Have Finally Unleashed AI on the World." *Wired*, Oct. 27, 2014.

Li, W., & Zhang, X. "Using Large Language Models for Automated Test Case Generation." *Journal of Software Testing and Verification*, 2022.

Maxwell, P. *AI for Quality Assurance: Leveraging Artificial Intelligence in Testing*. TechPress, 2021.

Johnson, J. *The AI-Driven Enterprise: How AI Can Transform Quality Assurance and Testing*. FutureTech Publishing, 2022.

Khair, M. A., Mallipeddi, S. R., & Varghese, A. "Software Testing in the Era of AI: Leveraging Machine Learning and Automation for Efficient Quality Assurance." *Asian Journal of Applied Science and Engineering*, Vol. 10, No. 4, December 2021, pp. 45-60.